PCI-SIG ENGINEERING DRAFT CHANGE NOTICE

TITLE:	Flattening Portal Bridge (FPB)
DATE:	Introduced: 19 April 2016 Updated: 20 Oct 2016 PWG Approval: Member Review – 20 Oct 2016
AFFECTED DOCUMENT:	PCI Express Base Specification, Rev. 3.0 / 3.1
SPONSOR:	Intel, Microsoft, NVidia

Part I

1. Summary of the Functional Changes

This ECR is intended to address a class of issues with PCI/PCIe architecture that relate to resource allocation inefficiency. To explain this, first we must define some terms:

- Static use cases, refer to scenarios where resources are allocated at system boot and then typically not changed again
- Dynamic use cases, refer to scenarios where run-time resource rebalancing (allocation of new resources, freeing of resources no longer needed) is required, due to hot add/remove, or by other needs.

In the Static cases there are limits on the size of hierarchies and number of Endpoints due to the Bus & Device Number "waste" caused by the PCI/PCIe architectural definition for Switches, and by the requirement that Downstream Ports associate an entire Bus Number with their Link. This proposal addresses this class of problems by "flattening" the use of Routing IDs so that Switches and Downstream Ports are able to make more efficient use of the available space.

For the Dynamic cases, until now, the "best known method" to avoid rebalancing has been to reserve large ranges of Bus Numbers and Memory Space in the Bridge above the relevant Endpoint(s) such that hopefully any needs can be satisfied within the pre-allocated ranges. This leads to additional waste, which makes the Routing ID issues worse, and this approach is difficult to implement in the general case, even for relatively simple cases, where, for example, one might have an SSD implementing a single Endpoint replaced by an unit that has a Switch, creating an internal hierarchy within the unit, so that although an initial allocation of just one Bus would have been sufficient, the initial allocation breaks immediately with the new unit.

For Memory Space the pre-allocation approach is problematic when hot-plugged Endpoints may require the allocation of Memory Space space below 4GB, which by its nature is a limited resource, which is quickly used up by pre-allocation of even relatively small amounts, and for which pre-allocation is unattractive because of the multiple system elements placing demands on system address space allocation below 4GB. Depending on multiple factors including a given system's physical memory addressing capability, there may in some cases also be resource constraints in Memory Space space above 4GB. Often, the constraints that apply to Memory Space below 4GB differ from those that apply above 4GB, and so this ECR provides separate mechanisms optimized for each.

This proposal addresses both the Static and Dynamic use cases by defining mechanisms to enable discontinuous resource range (re/)allocation for both Routing IDs and Memory Space. The intent is to allow system software the ability to maintain resource "pools" which can be allocated (and freed back to) at run-time, without disrupting other operations in progress as is required with rebalancing.

The Flattening Portal Bridge (FPB) is an optional Capability that may be implemented by Type 1 (Bridge) Functions in Root and Switch Ports to support more efficient and dense Routing ID allocation, and to enable reallocation of Routing ID resources without requiring the rebalancing of resources assigned elsewhere in a system, and to enable discontinuous Memory Space regions to avoid the need to rebalance Memory Space resources. IO space allocation is not modified by FPB as it is felt that the value is too low to justify the cost.

2. Benefits as a Result of the Changes

The benefits of implementing FPB are:

1. More efficient and dense allocation of Routing IDs, enabling larger hierarchies.
2. Runtime reallocation of resources for hot add/remove cases without the need to globally rebalance resources.
3. Removes the requirement for Routing ID's and Memory Space to be allocated in contiguous ranges.
4. Supports mixed systems including components that support FPB along with components that do not.
5. No changes to existing Discrete Endpoints

3. Assessment of the Impact

FPB will require new hardware and software, but has no effect unless enabled, and is disabled by default. Guidance is provided to support the coordinated introduction of hardware, firmware and system software supporting FPB.

Non-FPB RCs, Switches, Bridges and Endpoints can be used in mixed system environments along with RCs and Switches implementing FPB.

4. Analysis of the Hardware Implications

Hardware changes are required to implement the new FPB functionality. Only Type 1 functions implementing FPB are affected.

Endpoints and Type 0 functions are not affected.

As required to achieve the "Flattening" aspect of FPB, Upstream Ports can be mapped to Device Numbers other than 0. PCIe has always explicitly required non-ARI Devices to capture both the Bus and Device Numbers they are addressed with. However, because current hardware is not exposed to non-0 Device Numbers, there is the risk that not all existing hardware actually does what the spec requires, and we should try to flag non-compliant hardware ASAP. Devices supporting ARI will work with FPB, but require system software to confirgure FPB to assign the required block of Functions to the ARI device.

5. Analysis of the Software Implications

Software that intends to work with devices implementing the FPB functionality will be required to comprehend the new capability. Existing software will continue to function with FPB hardware, but will not be able to make use of FPB features.

6. Analysis of the C&I Test Implications

It is expected that HW implementing this will pass existing C & I tests. New C & I tests would be required if it is desirable to extend C & I coverage to explicitly evaluate FPB.

As noted above, it would be highly desirable to have PCISIG's C & I testing provide "FYI" evaluation ASAP of the requirement for non-ARI Functions to capture both Bus and Device Numbers, to identify non-compliant hardware as soon as possible.

Part II

Detailed Description of the change

Edit as shown in Section 2.2.8.1:

…

Table 2-20: Bridge Mapping for INTx Virtual Wires

~~Device Number for Device~~ Requester ID[7:3] from the Assert_INTx/Deassert_INTx Message received on Secondary Side of Bridge (Interrupt Source[*footnote*: The Requester ID of an Assert_INTx/Deassert_INTx Message will correspond to the Transmitter of the Message on that Link, and not necessarily to the original source of the interrupt.]) If ARI Forwarding is enabled, the value 0 must be used instead of Requester ID[7:3].	INTx Virtual Wire on Secondary Side of Bridge	Mapping to INTx Virtual Wire on Primary Side of Bridge
0,4,8,12,16,20,24,28	INTA	INTA
	INTB	INTB
	INTC	INTC
	INTD	INTD
1,5,9,13,17,21,25,29	INTA	INTB
	INTB	INTC
	INTC	INTD
	INTD	INTA
2,6,10,14,18,22,26,30	INTA	INTC
	INTB	INTD
	INTC	INTA
	INTD	INTB
3,7,11,15,19,23,27,31	INTA	INTD
	INTB	INTA
	INTC	INTB
	INTD	INTC

~~Note that the Requester ID of an Assert_INTx/Deassert_INTx Message will correspond to the Transmitter of the Message on that Link, and not necessarily to the original source of the interrupt.~~

…

[because the following material is all new material, it is not here marked as red+underline] Insert new section 6.x as follows:

6.x Flattening Portal Bridge (FPB)

6.x.1 Introduction

The Flattening Portal Bridge (FPB) is an optional mechanism which can be used to improve the scalability and runtime reallocation of Routing IDs and Memory Space resources.

For non-ARI Functions associated with an Upstream Port, the Routing ID consists of a 3 bit Function Number portion, which is determined by the construction of the Upstream Port hardware, and a 13 bit Bus Number and Device number portion, determined by the Downstream Port above the Upstream port.

For ARI Functions associated with an Upstream Port, the Routing ID consists of an 8 bit Function Number portion, and only the 8 bit Bus Number portion is determined by the Downstream Port above the Upstream port.

A Bridge that implements the FPB capability can itself also be referred to as an FPB. The FPB capability can be applied to any logical bridge, as illustrated in <Figure 6-x1>.

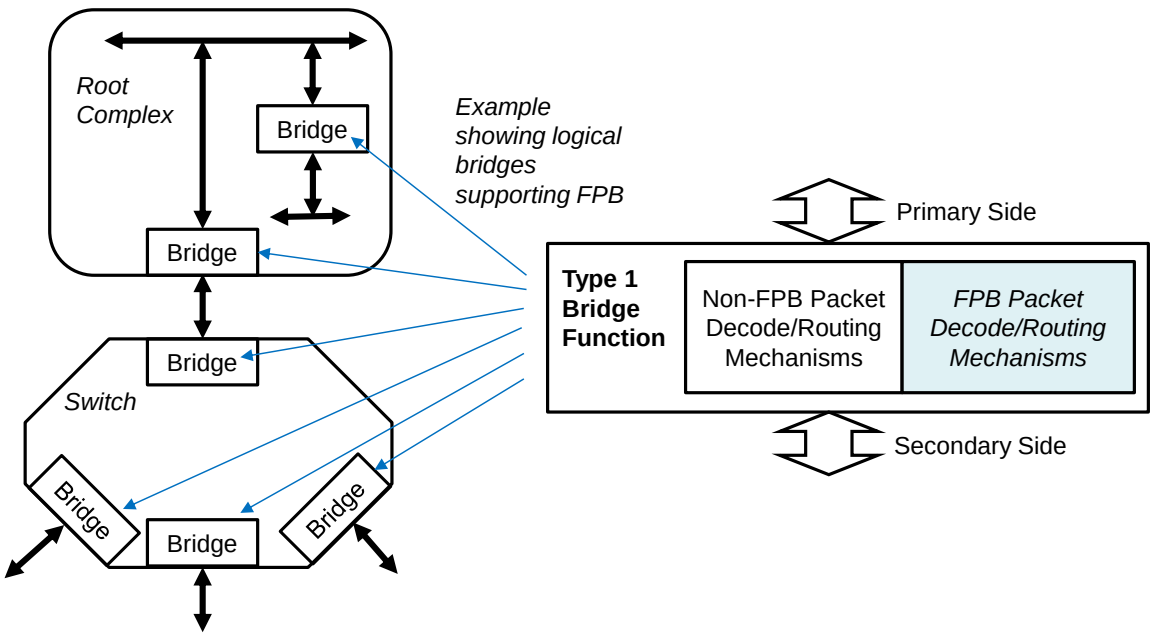

Figure 6-x1: FPB High Level Diagram and Example Topology

FPB changes the way Bus Numbers are consumed by Switches to reduce waste, by "flattening" the way Bus Numbers are used inside of Switches and by Downstream Ports (Figure 6-x1a).

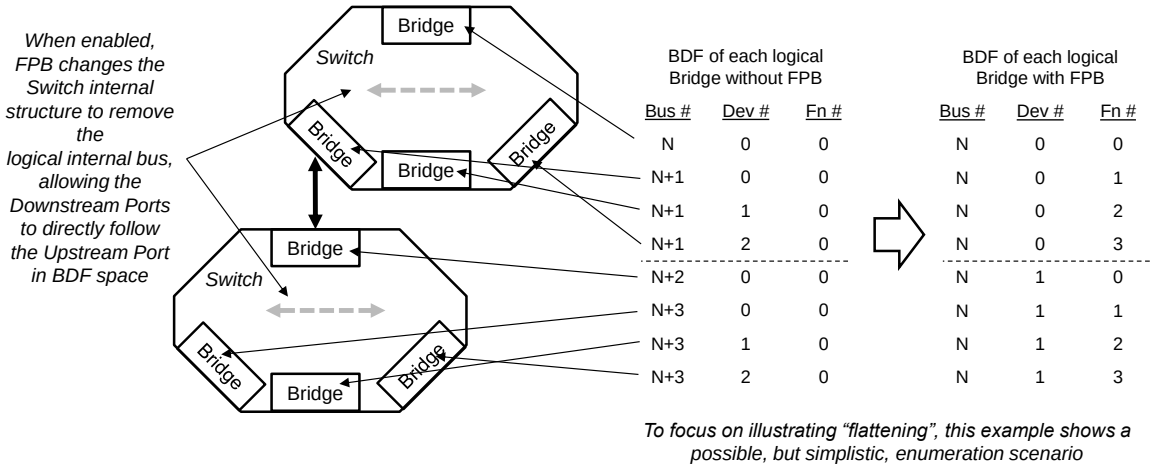

Figure 6-x1a: Example Illustrating "Flattening" of a Switch

FPB defines mechanisms for system software to allocate Routing IDs and Memory Space resources in non-contiguous ranges, enabling system software to assign pools of these resources from which it can allocate "bins" to Functions below the FPB. This is done using a bit vector where each bit when Set assigns a corresponding range of resources to the Secondary Side of the Bridge (see Figure 6-x1b).

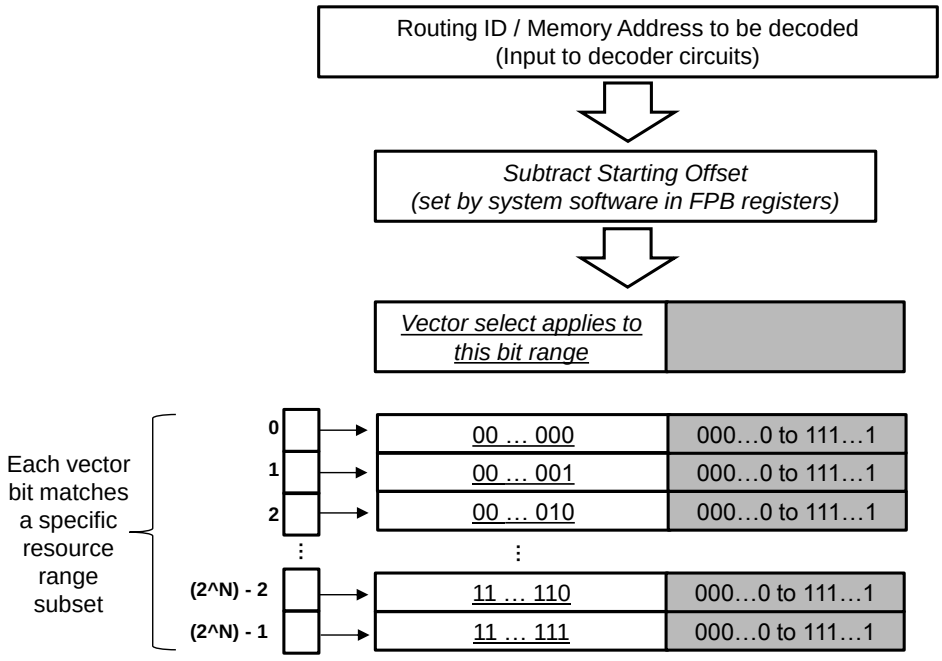

Figure 6-x1b: Vector Mechanism for Address Range Decoding

This allows system software to assign Routing IDs and/or Memory Space resources required by a device hot-add without having to rebalance other, already assigned resource ranges, and to return to the pool resources freed, for example by a hot remove event.

FPB is defined to allow both the non-FPB and FPB mechanisms to operate simultaneously, such that, for example, it is possible for system firmware/software to implement a policy where the non-FPB mechanisms continue to be used in parts of the system where the FPB mechanisms are not required (see Figure 6-x2). In this figure, the decode logic is assumed to provide a '1' output when a given TLP is decoded as being associated with the bridge's Secondary Side. The non-FPB decode mechanisms apply as without FPB, so for example only the Bus Number portion (bits 15:8) of a Routing ID is tested by the non-FPB decode logic when evaluating an ID routed TLP.

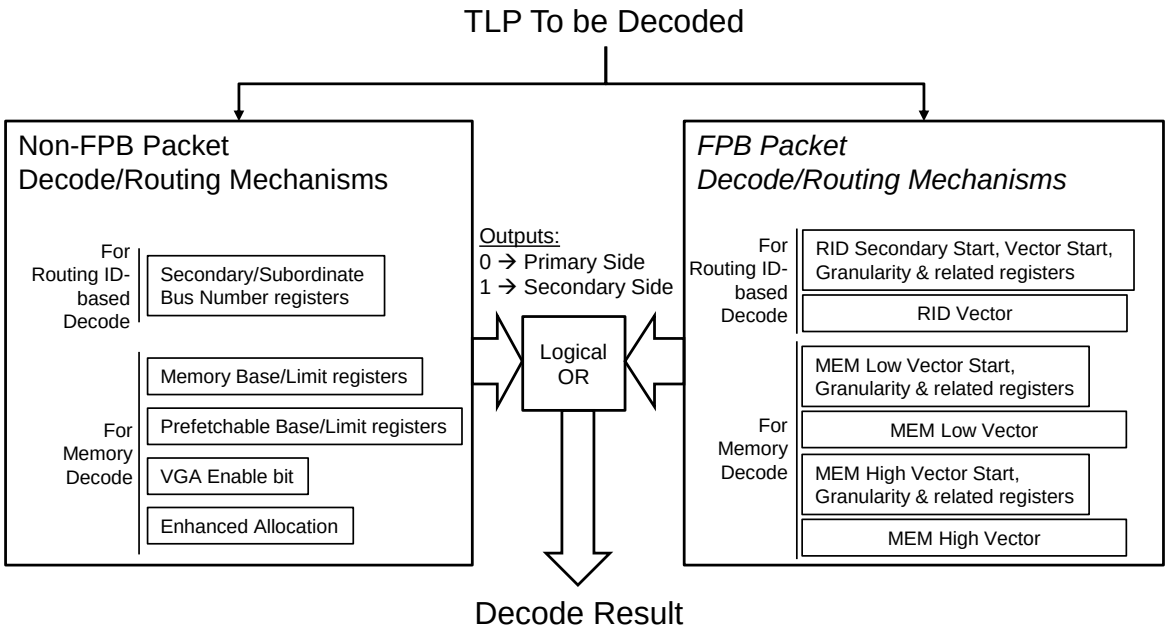

Figure 6-x2: Relationship between FPB and non-FPB Decode Mechanisms

It is important to recognize that, although FPB adds additional ways for a specific Bridge to decode a given TLP, FPB does not change anything about the fundamental ways that Bridges operate within the Switch and Root Complex architectural structures. FPB uses the same architectural concepts to provide management mechanisms for three different resource types:

1. Routing IDs
2. Memory below 4GB ("MEM Low")
3. Memory above 4GB ("MEM High")

A hardware implementation of FPB is permitted to support any combination of these three mechanisms. For each mechanism, FPB uses a bit-vector to indicate, for a specific subset range of the selected resource type, if resources within that range are associated with the Primary or Secondary side of the FPB. Hardware implementations are permitted to implement a small range of sizes for these vectors, and system firmware/software is

enabled to make the most effective use of the available vector by selecting an initial offset at which the vector is applied, and a granularity for the individual bits within the vector to indicate the size of the resource range to which the bits in a given vector apply.

6.x.2 Hardware and Software Requirements

The following rules apply when any of the FPB mechanisms are used:

- If system software violates any of the rules concerning FPB, the hardware behavior is undefined.
- It is permitted to implement FPB in any PCI Bridge (Type 1) Function, and every Function that implements FPB must implement the FPB Capability (see Section 7.y).
- If a Switch implements FPB then the Upstream Port and all Downstream Ports of the Switch must implement FPB.
- Software is permitted to enable FPB at some Switch Ports and not others.
- A Root Complex is permitted to implement FPB on some Root Ports but not on others.
- A Type 1 Function is permitted to implement the FPB mechanisms applying to any one, two or three of these elemental mechanisms:
    - Routing IDs (RID)
    - Memory below 4GB ("MEM Low")
    - Memory above 4GB ("MEM High")
- System software is permitted to enable any combination (including all or none) of the elemental mechanisms supported by a specific FPB.
- The error handling and reporting mechanisms, except where explicitly modified in this section, are unaffected by FPB.
- Following any reset of the FPB Function, the FPB hardware must Clear all bits in all implemented vectors.
- Once enabled (through the FPB RID Decode Mechanism Enable, FPB MEM Low Decode Mechanism Enable, and/or FPB MEM High Decode Mechanism Enable bits), if system software subsequently disables an FPB mechanism, the values of the entries in the associated vector are undefined, and if system software subsequently re-enables that FPB mechanism the FPB hardware must Clear all bits in the associated vector.
- If an FPB is implemented with the No_Soft_Reset bit Clear, when that FPB is cycled through D0→D3hot→D0, then all FPB mechanisms must be disabled, and the FPB must Clear all bits in all implemented vectors.
- If an FPB is implemented with the No_Soft_Reset bit Set, when that FPB is cycled through D0→D3hot→D0, then all FPB configuration state must not change, and the entries in the FPB vectors must be retained by hardware.
- Hardware is not required to perform any type of bounds checking on FPB calculations, and system software must ensure that the FPB parameters are correctly programmed
    - It is explicitly permitted for system software to program Vector Start values that cause the higher order bits of the corresponding vector to surpass the resource range associated with a given FPB, but in these cases system software must ensure that those higher order bits of the vector are Clear.
    - Examples of errors that system software must avoid include duplication of resource allocation, combinations of start offsets with set vector bits that could create "wrap-around" or bounds errors

The following rules apply to the FPB Routing ID (RID) mechanism:

- FPB hardware must consider a specific range of RIDs to be associated with the Secondary side of the FPB if the Bus Number portion falls within the Bus Number range indicated by the values programmed in the Secondary and Subordinate Bus Number registers logically OR'd with the value programmed into the corresponding entry in the RID Vector.
- If it is intended to use only the FPB RID mechanism for BDF decoding, then system software must ensure that both the Secondary and Subordinate Bus Number registers are 0.
- System software must ensure that the FPB routing mechanisms are configured such that Configuration Requests targeting Functions Secondary side of the FPB will be routed by the FPB from the Primary to Secondary side of the FPB.

When ARI is not enabled, The FPB RID mechanism can be applied with different granularities, programmable by system software through the FPB RID Vector Granularity register in the FPB RID Vector Control 1 Register. Figure 6-x3 illustrates the relationships between the layout of RIDs and the supported granularities. The reader may find it helpful to refer to this figure when considering the requirements defined below and in the definition of the Flattening Portal Bridge (FPB) Capability (Sect. 7.y).

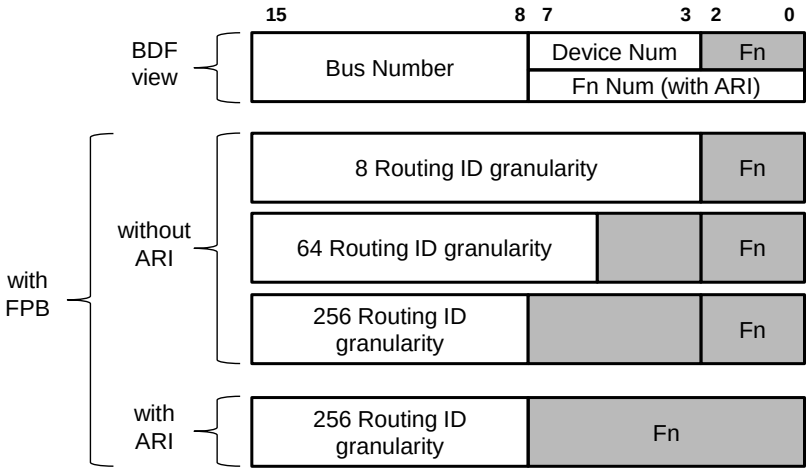

Figure 6-x3: Routing IDs (RIDs) and Supported Granularities

- System software must program the FPB RID Vector Granularity and FPB RID Vector Start fields in the FPB RID Vector Control 1 register per the constraints described in the descriptions of those fields.
- For all FPBs other than those associated with Upstream Ports of Switches:
    - When ARI Forwarding is not supported, or when the ARI Forwarding Enable bit in the Device Control 2 register is Clear, FPB hardware must convert a Type 1 Configuration Request received on the Primary side of the FPB to a Type 0 Configuration Request on the Secondary side of the FPB when bits 15:3 of the Routing ID of the Type 1 Configuration Request matches the value in the RID Secondary Start field in the FPB RID Vector Control 2 Register, and system software must configure the FPB accordingly.

  - When the ARI Forwarding Enable bit in the Device Control 2 register is Set, FPB hardware must convert a Type 1 Configuration Request received on the Primary side of the FPB to a Type 0 Configuration Request on the Secondary side of the FPB when the Bus Number portion of the Routing ID of the Type 1 Configuration Request matches the value in the Bus Number address (bits 15:8 only) of the Secondary Start field in the FPB RID Vector Control 2 Register, and system software must configure the FPB accordingly.
- For FPBs associated with Upstream Ports of Switches only, when the FPB RID Decode Mechanism Enable bit is Set, FPB hardware must use the FPB Num Sec Dev field of the FPB Capability Register to indicate the quantity of Device Numbers associated with the Secondary Side of the Upstream Port Bridge, which must be used by the FPB in addition to the RID Secondary Start field in the FPB RID Vector Control 2 Register to determine when a Configuration Request received on the Primary side of the FPB targets one of the Downstream Ports of the Switch, determining in effect when such a Request must be converted form a Type 1 Configuration Request to a Type 0 Configuration Request, and system software must configure the FPB appropriately.
  - System software configuring FPB must comprehend that the logical internal structure of a Switch will change depending on the value of the FPB RID Decode Mechanism Enable bit in the Upstream Port of a Switch.
  - Downstream Ports must use their corresponding RID values, and their Requester IDs and Completer IDs, as determined by the Upstream Port's FPB Num Sec Dev and RID Secondary Start values
- FPB's must implement bridge mapping for INTx virtual wires (see Sect. 2.2.8.1)
- Hardware and software must apply this algorithm (or the logical equivalent) to determine which entry in the FPB RID Vector applies to a given Routing ID (RID) address:
  - IF the RID is below the value of FPB RID Vector Start, then the RID is out of range (below the start) and so cannot be associated with the Secondary side of the bridge, ELSE
  - calculate the offset within the vector by first subtracting the value of FPB RID Vector Start, then dividing this according to the value of FPB RID Vector Granularity to determine the bit index within the vector.
  - IF the bit index value is greater than the length indicated by FPB RID Vector Size Supported, then the RID is out of range (beyond the top of the range covered by the vector) and so cannot be associated with the Secondary side of the bridge, ELSE
  - if the bit value within the vector at the calculated bit index location is 1b, THEN the RID address is associated with the Secondary side of the bridge, ELSE the RID address is associated with the Primary side of the bridge.

The following rules apply to the MEM Low mechanism:

The FPB MEM Low mechanism can be applied with different granularities, programmable by system software through the FPB MEM Low Vector Granularity register in the FPB MEM Low Vector Control Register. Figure 6-x4 illustrates the relationships between the layout of addresses in the memory address space below 4GB to which the FPB MEM Low mechanism applies. The reader may find it helpful to refer to this figure when considering the requirements defined below and in the definition of the Flattening Portal Bridge (FPB) Capability (Sect. 7.y).

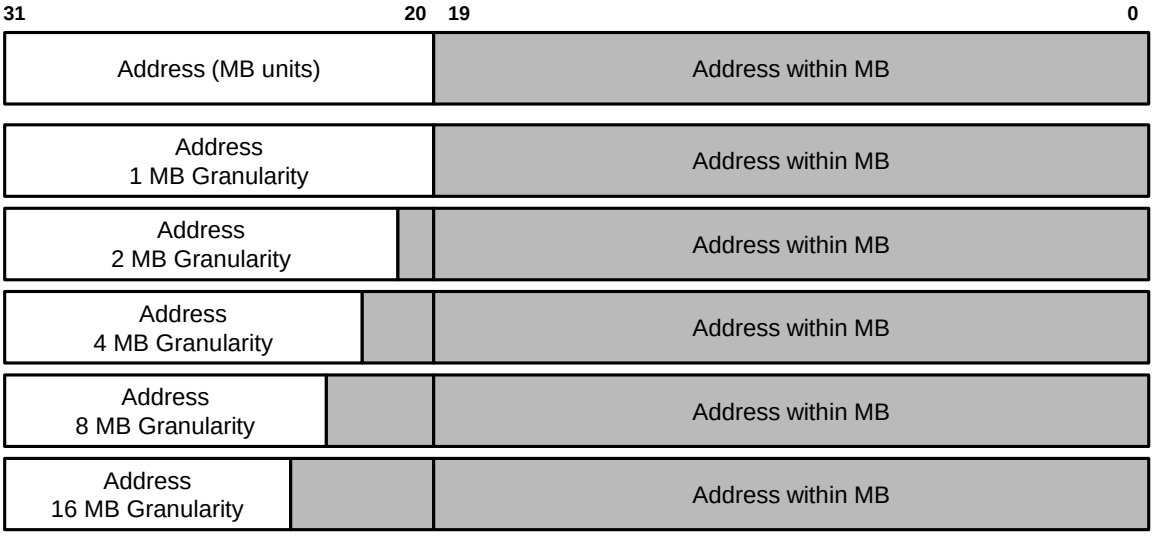

Figure 6-x2: Addresses in Memory Below 4GB and Effect of Granularity

- System software must program the FPB MEM Low Vector Granularity and FPB MEM Low Vector Start fields in the FPB MEM Low Vector Control register per the constraints described in the descriptions of those fields.
- FPB hardware must consider a specific Memory address to be associated with the Secondary side of the FPB if that Memory address falls within any of the ranges indicated by the values programmed in other Bridge Memory decode registers (enumerated below) logically OR'd with the value programmed into the corresponding entry in the MEM Low Vector. Other Bridge Memory decode registers include:
    - Memory Base/Limit registers in the Type 1 (Bridge) header
    - Prefetchable Base/Limit registers in the Type 1 (Bridge) header
    - VGA Enable bit in the Bridge Control Register of the Type 1 (Bridge) header
    - Enhanced Allocation (EA) Capability
    - FPB MEM High mechanism (if supported and enabled)
- Hardware and software must apply this algorithm (or the logical equivalent) to determine which entry in the FPB MEM Low Vector applies to a given Memory address:
    - If the Memory address is below the value of FPB MEM Low Vector Start, then the Memory address is out of range (below) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
    - calculate the offset within the vector by first subtracting the value of FPB MEM Low Vector Start, then dividing this according to the value of FPB MEM Low Vector Granularity to determine the bit index within the vector.
    - If the bit index value is greater than the length indicated by FPB MEM Low Vector Size Supported, then the Memory address is out of range (above) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
    - if the bit value within the vector at the calculated bit index location is 1b, then the Memory address is associated with the Secondary side of the bridge, else the Memory address is associated with the Primary side of the bridge.

The following rules apply to the MEM High mechanism:

- System software must program the FPB MEM High Vector Granularity and FPB MEM High Vector Start Lower fields in the FPB MEM High Vector Control 1 register per the constraints described in the descriptions of those fields.
- FPB hardware must consider a specific Memory address to be associated with the Secondary side of the FPB if that Memory address falls within any of the ranges indicated by the values programmed in other Bridge Memory decode registers (enumerated below) logically OR'd with the value programmed into the corresponding entry in the MEM Low Vector. Other Bridge Memory decode registers include:
    - Memory Base/Limit registers in the Type 1 (Bridge) header
    - Prefetchable Base/Limit registers in the Type 1 (Bridge) header
    - VGA Enable bit in the Bridge Control Register of the Type 1 (Bridge) header
    - Enhanced Allocation (EA) Capability
    - FPB MEM Low mechanism (if supported and enabled)
- Hardware and software must apply this algorithm to determine which entry in the FPB MEM High Vector applies to a given Memory address:
    - If the Memory address is below the value of FPB MEM High Vector Start, then the Memory address is out of range (below) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
    - calculate the offset within the vector by first subtracting the value of FPB MEM High Vector Start, then dividing this according to the value of FPB MEM High Vector Granularity to determine the bit index within the vector.
    - If the bit index value is greater than the length indicated by FPB MEM High Vector Size Supported, then the Memory address is out of range (above) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
    - if the bit value within the vector at the calculated bit index location is 1b, then the Memory address is associated with the Secondary side of the bridge, else the Memory address is associated with the Primary side of the bridge.

IMPLEMENTATION NOTE

FPB Address Decoding

FPB uses a bit vector mechanism to decode ranges of Routing IDs, and Memory Addresses above and below 4GB. A bridge supporting FPB contains the following for each resource type/range where it supports the use of FPB:

- A Bit vector
- A Start Address register
- A Granularity register

These are used by the bridge to determine if a given address is part of the range decoded by FPB as associated with the secondary side of the bridge. An address that does not determined to be associated with the secondary side of the bridge using either or both of the non-FPB decode mechanisms and the FPB decode mechanisms is (by default) associated with the primary side of the bridge. Here, when we use the term "associated" we mean, for example, that the bridge will apply the following handling to TLPs:

- Associated with Primary, Received at Primary → Unsupported Request (UR)

- Associated with Primary, Received at Secondary → Forward upstream
- Associated with Secondary, Received at Primary → Forward downstream
- Associated with Secondary, Received at Secondary → Unsupported Request (UR)

In FPB, every bit in the vector represents a range of resources, where the size of that range is determined by the selected granularity. If a bit in the vector is Set, it indicates that TLPs addressed to an address within the corresponding range are to be associated with the secondary side of the bridge. The specific range of resources each bit represents is dependent on the index of that bit, and the values in the Start Address & Granularity registers. The Start Address register indicates the lowest address described by the bit vector. The Granularity register indicates the size of the region that is represented by each bit. Each successive bit in the vector applies to the subsequent range, increasing with each bit according to the Granularity.

For example, consider a bridge using FPB to describe a MEM Low range. FPB MEM Low Vector Start has been set to 0xFC0, indicating that the range described by the bit vector starts at address FC000000h. FPB MEM Low Vector Granularity has been set to 0000b, indicating that each bit represents a 1 MB range.

From these values we can determine that bit 0 of the vector represents a 1MB range starting at FC0000000h (FC000000h-FC0FFFFFh), bit 1 represents FC100000h-FC1FFFFFh, etc.

Bits in the vector that are set to 0 indicate that the range is not included in the range described by FPB. In the above example, If bit 0 is Clear, packets addressed to anywhere between FC000000h and FC0FFFFFh should not be routed to the secondary bus of the bridge due to FPB.

IMPLEMENTATION NOTE

Hardware and Software Considerations for FPB

FPB is intended to address a class of issues with PCI/PCIe architecture that relate to resource allocation inefficiency. These issues can be categorized as “static” or “dynamic” use case scenarios, where static use cases refer to scenarios where resources are allocated at system boot and then typically not changed again, and dynamic use cases refer to scenarios where run-time resource rebalancing (allocation of new resources, freeing of resources no longer needed) is required, due to hot add/remove, or by other needs.

In the Static cases there are limits on the size of hierarchies and number of Endpoints due to the Bus & Device Number “waste” caused by the PCI/PCIe architectural definition for Switches and Downstream Ports. FPB addresses this class of problems by “flattening” the use of Routing IDs (RIDs) so that Switches and Downstream Ports are able to make more efficient use of the available space.

For the Dynamic cases, without FPB, the “best known method” to avoid rebalancing has been to reserve large ranges of Bus Numbers and Memory Space in the Bridge above the relevant Port or Endpoint such that hopefully any future needs can be satisfied within the pre-allocated ranges. This leads to additional waste, which makes the Routing ID issues worse, and this approach is difficult to implement in the general case, even for relatively simple cases, where, for example, one might have an SSD implementing a single Endpoint replaced by an unit that has a Switch with internal hierarchy, so that although an initial allocation of just one Bus would have been sufficient, the initial allocation breaks immediately with the new unit.

For Memory Space the pre-allocation approach is problematic when hot-plugged Endpoints may require the allocation of Memory Space below 4GB, which by its nature is a limited resource, which is quickly used up by pre-allocation of even relatively small amounts, and for which pre-allocation is unattractive because of the multiple system elements placing demands on system address space allocation below 4GB.

FPB includes mechanisms to enable discontinuous resource range (re/)allocation for both Requester IDs and Memory Space. The intent is to allow system software the ability to maintain resource “pools” which can be allocated (and freed back to) at run-time, without disrupting other operations in progress as is required with rebalancing.

To support the run time use of FPB by system software, FPB hardware implementations should avoid introducing stalls or other types of disruptions to transactions in flight, including during the times that system software is modifying the state of the FPB hardware. It is not, however, expected that hardware will attempt to identify cases where system software erroneously modifies the FPB configuration in a way that does affect transactions in flight. Just as with the non-FPB mechanisms, it is the responsibility of system software to ensure that system operation is not corrupted due to a reconfiguration operation.

It is not explicitly required that system firmware/software perform the enabling and/or disabling of FPB mechanisms in a particular sequence, however care should be taken to implement resource allocation operations in a hierarchy such that the hardware and software elements of the system are not corrupted or caused to fail.

Edit as shown:

6.12.1.1 *ACS Downstream Ports*

This section applies to Root Ports and Downstream Switch Ports that implement an ACS Extended Capability structure. This section applies to Downstream Port Functions both for single-Function devices and multi-Function devices.

- ACS Source Validation: must be implemented.

  When enabled, the Downstream Port tests the Bus Number from the Requester ID of each Upstream Request received by the Port to determine if it is associated with the Secondary side of the virtual Bridge associated with the Downstream Port, by either or both of:

  - Determining that the Requester ID falls within the Bus Number "aperture" of the Port – the inclusive range specified by the Secondary Bus Number register and the Subordinate Bus Number register
  - If FPB is implemented and enabled, determing that the Requester ID is associated with the bridge's Secondary Side by the application of the FPB Routing ID mechanism.

  If the Bus Number from the Requester ID of the Request is not within this aperture, this is a reported error (ACS Violation) associated with the Receiving Port (see Section 6.12.4.)

…

7.3 Configuration Transaction Rules

7.3.1 Device Number

…

Except when FPB Routing ID Mechanisms are used (see Sect 6.x), Downstream Ports that do not have ARI Forwarding enabled must associate only Device 0 with the device attached to the Logical Bus representing the Link from the Port. Configuration Requests targeting the Bus Number associated with a Link specifying Device Number 0 are delivered to the device attached to the Link; Configuration Requests specifying all other Device Numbers (1-31) must be terminated by the Switch Downstream Port or the Root Port with an Unsupported Request Completion Status (equivalent to Master Abort in PCI).

[[add paragraph break here]] Non-ARI Devices must not assume that Device Number 0 is associated with their Upstream Port, but must capture their assigned Device Number as discussed in Section 2.2.6.2. Non-ARI Devices must respond to all Type 0 Configuration Read Requests, regardless of the Device Number specified in the Request.

Switches, and components wishing to incorporate more than eight Functions at their Upstream Port, are permitted to implement one or more "virtual switches" represented by multiple Type 1 (PCI-PCI Bridge) Configuration Space headers as illustrated in Figure 7-2. These virtual switches serve to allow fan-out beyond eight Functions. FPB provides a "flattening" mechanism that, when enabled, causes the virtual Bridges of the Downstream Ports to appear in configuration space at successive RID addresses following the RID of the Upstream Port (see 6.x).

…

7.3.2 Configuration Transaction Addressing

…

 Device Number – Device Number association is discussed in Section 7.3.1, and in Section 6.x. When an ARI Device is targeted and the Downstream Port immediately above it is enabled for ARI Forwarding, the Device Number is implied to be 0, and the traditional Device Number part of the Routing ID is used instead as part of an 8-bit Function Number field. See Section 6.13.

…

7.3.3 Configuration Request Routing Rules

…

 If Configuration Request Type is 1, apply the following tests, in sequence, to the Bus Number and Device Number fields:

- o If in the case of a PCI Express-PCI Bridge, equal to the Bus Number assigned to secondary PCI bus or, in the case of a Switch or Root Complex, equal to the Bus Number and decoded Device Numbers assigned to one of the Root (Root Complex) or Downstream Ports (Switch), or if required based on the FPB Routing ID mechanism,
  - Transform the Request to Type 0 by changing the value in the Type[4:0] field of the Request (see Table 2-3) – all other fields of the Request remain unchanged
  - Forward the Request to that Downstream Port (or PCI bus, in the case of a PCI Express-PCI Bridge)
- o If not equal to the Bus Number of any of Downstream Ports or secondary PCI bus, but in the range of Bus Numbers assigned to either a Downstream Port or a secondary PCI bus, or if required based on the FPB Routing ID mechanism,
  - Forward the Request to that Downstream Port interface without modification
- o Else (none of the above)
  - The Request is invalid – follow the rules for handling Unsupported Requests

[because the following material is all new material, it is not here marked as red+underline] Insert new section 7.y following Enhanced Allocation and ahead of PCI Express Capability Structure:

7.y Flattening Portal Bridge (FPB) Capability

The Flattening Portal Bridge (FPB) Capability is an optional Capability that is required for any Bridge Function that implements FPB. The FPB Capability structure is defined in this section.

If a Switch implements FPB then the Upstream Port and all Downstream Ports of the Switch must implement the FPB Capability Structure. A Root Complex is permitted to implement the FPB Capability Structure on some Root Ports but not on others. A Root Complex is permitted to implement the FPB Capability for internal logical busses.

7.y.1 FPB Capability Header (Offset 00h)

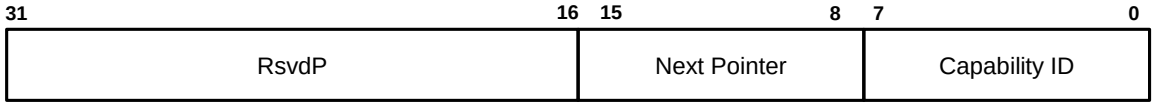

Figure 7-y1: FPB Capability Header

Table 7-y1: FPB Capability Header

Bit Location	Register Description	Attributes
7:0	**CAP_ID** – Must be set to 15h	RO
15:8	**NXT_PTR** - Pointer to the next item in the capabilities list. Must be NULL for the final item in the list.	RO
31:16	**Reserved**	RsvdP

7.y.2 FPB Capability Register (Offset 04h)

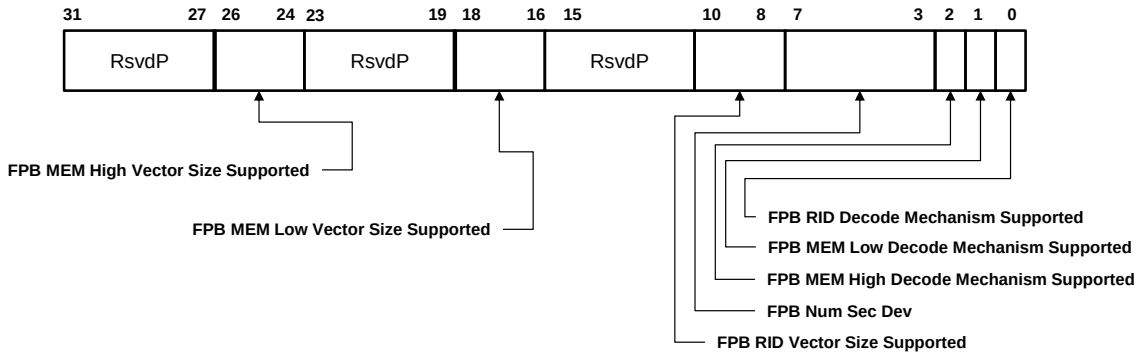

Figure 7-y2: FPB Capability Register

Table 7-y2: FPB Capability Register

Bit Location	Register Description	Attributes
0	**FPB RID Decode Mechanism Supported** – If Set, indicates that the RID Vector mechanism is supported.	RO
1	**FPB MEM Low Decode Mechanism Supported** - If Set, indicates that the MEM Low Vector mechanism is supported.	RO
2	**FPB MEM High Decode Mechanism Supported** - If Set, indicates that the Mem High mechanism is supported.	RO
7:3	**FPB Num Sec Dev** - For Upstream Ports of Switches only, this field indicates the quantity of Device Numbers associated with the Secondary Side of the Upstream Port Bridge. The quantity is determined by adding one to the numerical value of this field. Although it is encouraged that Switch implementations assign Downstream Ports using all 8 allowed Functions per allocated Device Number, such that all Downstream Ports are assigned within a contiguous range of Device and Function Numbers, it is, however, explicitly permitted to assign Downstream Ports to Function Numbers that are not contiguous within the indicated range of Device Numbers, and system software is required to scan for Downstream Port Bridges at every Function Number within the indicated quantity of Device Numbers associated with the Secondary Side of the Upstream Port. This field is Reserved for Downstream Ports.	RO/RsvdP
10:8	**FPB RID Vector Size Supported** – Indicates the size of the FPB RID Vector implemented in hardware, and constrains the allowed values software is permitted to write to the FPB RID Vector Granularity field. Defined encodings are: Value Size Allowed Granularities in RID units 000b 256 bits 8, 64, 256 010b 1K bits 8, 64 101b 8K bits 8 All other encodings are Reserved If the FPB RID Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.	RO
15:11	Reserved	RsvdP

18:16	**FPB MEM Low Vector Size Supported** – Indicates the size of the Mem Low Vector implemented in hardware, and constrains the allowed values software is permitted to write to the FPB MEM Low Vector Start field. Defined encodings are: Value Size Allowed Granularities in MB units 000b 256 bits 1, 2, 4, 8, 16 001b 512 bits 1, 2, 4, 8 010b 1K bits 1, 2, 4 011b 2K bits 1, 2 100b 4K bits 1 All other encodings are Reserved If the FPB Mem Low Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.	RO
23:19	Reserved	RsvdP
26:24	**FPB MEM High Vector Size Supported** – Indicates the size of the Mem Low Vector implemented in hardware. Defined encodings are: 000b 256 bits 001b 512 bits 010b 1K bits 011b 2K bits 100b 4K bits 101b 8K bits All other encodings are Reserved All defined Granularities are allowed for all defined vector sizes. If the FPB Mem High Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.	RO
31:27	Reserved	RsvdP

7.y.3 FPB RID Vector Control 1 Register (Offset 08h)

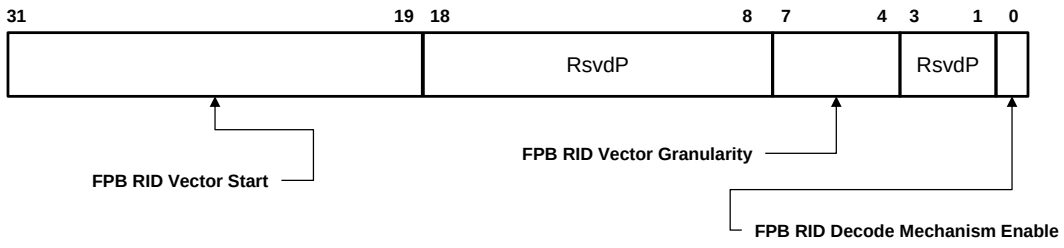

Figure 7-y3: FPB RID Vector Control 1 Register

Table 7-y3: FPB RID Vector Control 1 Register

Bit Location	Register Description	Attributes
0	**FPB RID Decode Mechanism Enable** – When Set, enables the FPB RID Decode Mechanism If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.	RW/RO
3:1	Reserved	RsvdP
7:4	**FPB RID Vector Granularity** – The value written by software to this field controls the granularity of the FPB RID Vector and the required alignment of the FPB RID Vector Start field (below). Defined encodings are: Value Granularity 0000b 8 RIDs 0011b 64 RIDs 0101b 256 RIDs All other encodings are Reserved Based on the implemented FPB RID Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. If the ARI Forwarding Enable bit in the Device Control 2 Register is Set, then software must program 0101b into this field, if this field is programmable. Default value for this field is 0000b.	RW/RO
18:8	Reserved	RsvdP

31:19	**FPB RID Vector Start** – The value written by software to this field controls the offset at which the FPB RID Vector is applied. The value represents a RID offset in units of 8 RIDs, such that bit 0 of the FPB RID Vector represents the range of RIDs starting from the value represented this register up to that value plus the FPB RID Vector Granularity minus 1, and bit 1 represents range from this register value plus granularity up to that value plus FPB RID Vector Granularity minus 1, etc. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB RID Vector Granularity Field as indicated here: FPB RID Vector Granularity Start Alignment Constraint 0000b <no constraint> 0011b …000b 0101b …00000b All other encodings are Reserved If this requirement is violated, the hardware behavior is undefined. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. If the ARI Forwarding Enable bit in the Device Control 2 Register is Set, then software must program 1111_1111_0000_0b into this field, and the hardware behavior is undefined if any other value is programmed. Default value for this field is 0000_0000_0000_0b.	RW/RO

7.y.4 FPB RID Vector Control 2 Register (Offset 0Ch)

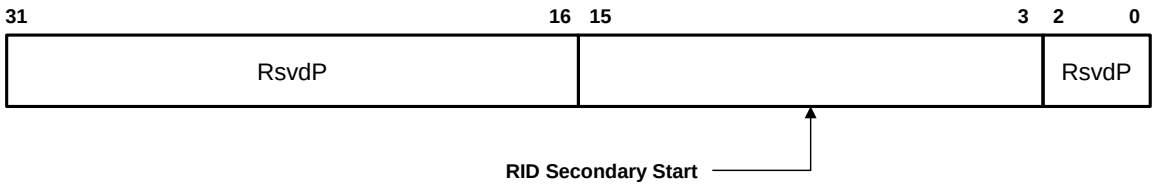

Figure 7-y4: FPB RID Vector Control 2 Register

Table 7-y4: FPB RID Vector Control 2 Register

Bit Location	Register Description	Attributes
2:0	Reserved	RsvdP
15:3	**RID Secondary Start** – The value written by software to this field controls the RID offset at which Type 1 Configuration Requests passing downstream through the bridge must be converted to Type 0. Bits[2:0] of the RID offset are fixed by hardware as 000b and cannot be modified. When the ARI Forwarding Enable bit in the Device Control 2 register is Set, then software must write bits 7:3 of this field to 00000b. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000_0000_0000_0b.	RW/RO
31:16	Reserved	RsvdP

7.y.5 FPB MEM Low Vector Control Register (Offset 10h)

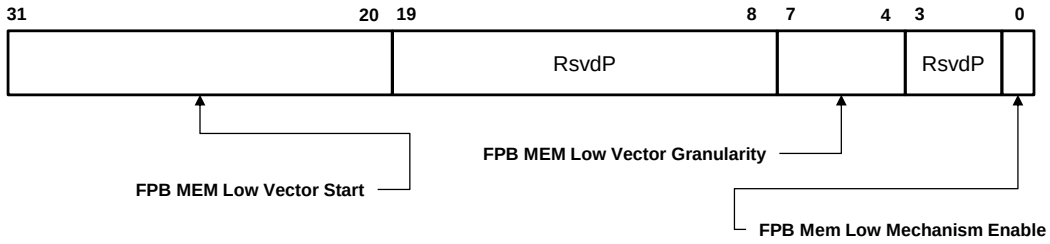

Figure 7-y5: FPB MEM Low Vector Control Register

Table 7-y5: FPB MEM Low Vector Control Register

Bit Location	Register Description	Attributes
0	**FPB MEM Low Decode Mechanism Enable** - When Set, enables the FPB MEM Low Decode mechanism. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.	RW/RO
3:1	Reserved	RsvdP

7:4	**FPB MEM Low Vector Granularity** – The value written by software to this field controls the granularity of the FPB MEM Low Vector, and the required alignment of the FPB MEM Low Vector Start field (below). Defined encodings are: Value Granularity 0000b 1MB 0001b 2MB 0010b 4MB 0011b 8MB 0100b 16MB All other encodings are Reserved Based on the implemented FPB MEM Low Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000b.	RW/RO
19:8	Reserved	RsvdP
31:20	**FPB MEM Low Vector Start** – The value written by software to this field sets the base address at which the FPB MEM Low Vector is applied. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM Low Vector Granularity Field as indicated here: FPB MEM Low Vector Granularity Constraint 0000b <no constraint> 0001b …0b 0010b …00b 0011b …000b 0100b …0000b If this requirement is violated, the hardware behavior is undefined. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 000h.	RW/RO

7.y.6 FPB MEM High Vector Control 1 Register (Offset 14h)

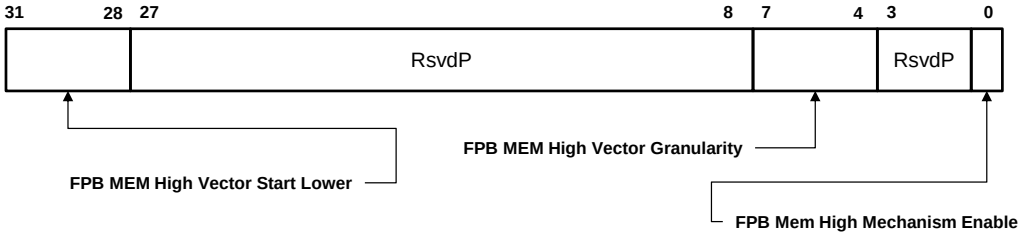

Figure 7-y6: FPB MEM High Vector Control 1 Register

Table 7-y6: FPB MEM High Vector Control 1 Register

Bit Location	Register Description	Attributes
0	**FPB MEM High Decode Mechanism Enable** - When Set, enables the FPB MEM High Decode mechanism. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.	RW/RO
3:1	Reserved	RsvdP

7:4	**FPB MEM High Vector Granularity** – The value written by software to this field controls the granularity of the FPB MEM High Vector, and the required alignment of the FPB MEM High Vector Start Lower field (below). Software is permitted to select any allowed Granularity from the table below regardless of the value in the FPB MEM High Vector Size Supported field. Defined encodings are: Value Granularity 0000b 256MB 0001b 512MB 0010b 1GB 0011b 2GB 0100b 4GB 0101b 8GB 0110b 16GB 0111b 32GB Based on the implemented FPB MEM High Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000b.	RW/RO
27:8	Reserved	RsvdP

31:28	**FPB MEM High Vector Start Lower** – The value written by software to this field sets the lower bits of the base address at which the FPB MEM High Vector is applied. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM High Vector Granularity Field as indicated here: FPB MEM High Vector Granularity — Constraint 0000b — <no constraint> 0001b — …0b 0010b — …00b 0011b — …000b 0100b — …0000b 0101b — …00000b 0110b — …000000b 0111b — …0000000b If this requirement is violated, the hardware behavior is undefined. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0h.	RW/RO

7.y.7 FPB MEM High Vector Control 2 Register (Offset 18h)

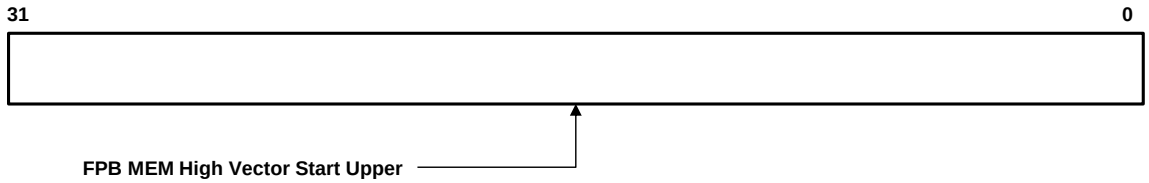

Figure 7-y7: FPB MEM High Vector Control 2 Register

Table 7-y7: FPB MEM High Vector Control 2 Register

Bit Location	Register Description	Attributes

31:0	**FPB MEM High Vector Start Upper** – The value written by software to this field indicates bits 63:32 of the base address at which the FPB MEM High Vector is applied. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000_0000h.	RW/RO

7.y.8 FPB Vector Access Control Register (Offset 1Ch)

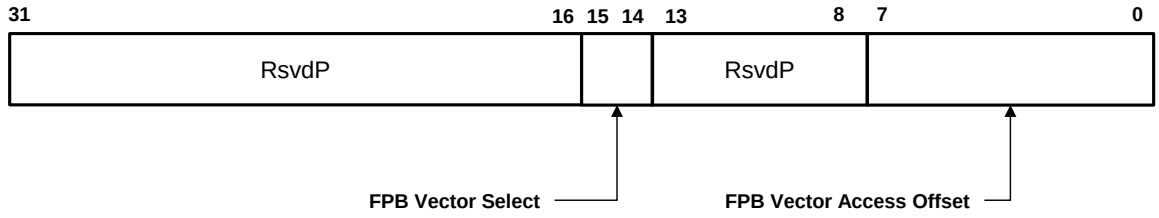

Figure 7-y8: FPB Vector Access Control Register

Table 7-y8: FPB Vector Access Control Register

Bit Location	Register Description	Attributes

7:0	**FPB Vector Access Offset** – The value in this field indicates the offset of the 32b portion of the FPB BD, MEM Low or MEM High, Vector that can be read or written by means of the FPB Vector Access Data Register. The selection of BD, MEM Low or MEM High is made by the value written to the FPB Vector Select field of this register. The bits of this field map to the offset according to the value in the corresponding FPB Vector Size Supported field as shown here: Offset Bits This Field 000b 2:0 2:0 (7:3 unused) 001b 3:0 3:0 (7:4 unused) 010b 4:0 4:0 (7:5 unused) 011b 5:0 5:0 (7:6 unused) 100b 6:0 6:0 (7 unused) 101b 7:0 7:0 All other encodings are Reserved Bits in this field that are unused per the table above must be written by software as 0b, and are permitted but not required to be implemented as RO. Default value for this field is 00h	RW/RO
13:8	Reserved	RsvdP
15:14	**FPB Vector Select** – The value written to this field selects the Vector to be accessed at the indicated FPB Vector Access Offset, encoded as: 00: BD 01: MEM Low 10: MEM High 11: Reserved Default value for this field is 00b	RW
31:16	Reserved	RsvdP

7.y.9 FPB Vector Access Data Register (Offset 20h)

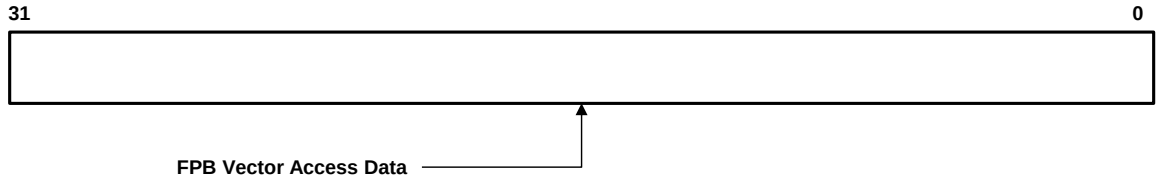

Figure 7-y9: FPB Vector Access Data Register

Table 7-y9: FPB Vector Access Data Register

Bit Location	Register Description	Attributes
31:0	**FPB Vector Access Data** – Reads from this register return the DW of data from the FPB Vector at the location determined by the value in the FPB Vector Access Offset Register. Writes to this register replace the DW of data from the FPB Vector at the location determined by the value in the FPB Vector Access Offset Register. Default value for this field is 0000h	RW

<end>